Wisdom of the Machines: Federated Learning using OPAL

by

Abdulrahman Alotaibi

B.S., Old Dominion University (2013)

Submitted to the Program in Media Arts and Sciences
in partial fulfillment of the requirements for the degree of

Master of Science in Media Arts and Sciences

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2018

Signature redacted

Author .....
Program in Media Arts and Sciences
September, 2018

Signature redacted

Certified by .....
Alex 'Sandy' Pentland
Toshiba Professor of Media Arts and Sciences
Thesis Supervisor

Signature redacted

Accepted by .....
Tod Machover
Academic Head, Program in Media Arts and Sciences

Wisdom of the Machines: Federated Learning using OPAL

by

Abdulrahman Alotaibi

Submitted to the Program in Media Arts and Sciences
on September, 2018, in partial fulfillment of the
requirements for the degree of
Master of Science in Media Arts and Sciences

Abstract

Wisdom of the crowds (WOC) is an old concept that started by recording and aggregating people's estimations. It is one of the useful tools that exists today and allows many estimation applications to work correctly. Moreover, Open algorithms (OPAL) is a useful platform that enables institutions and individuals to share sensitive data, and increases the privacy of the data. In addition, federated learning is a new way to build and generate machine learning models by aggregating their hyperparameters. In this thesis, I show how to combine the three different concepts to build machine learning models on top of OPAL that utilize federated learning on a network. I then extend OPAL to support this new feature and demonstrate how to build a machine learning model using small independent models.

Thesis Supervisor: Alex 'Sandy' Pentland
Title: Toshiba Professor of Media Arts and Sciences

Acknowledgments

This is the hardest part of my thesis to write because I have received help from many people, and I am afraid to forget to list everyone. I will try my best to list all the people that I can remember and I seek forgiveness if I left your name out.

First of all, I would like to start by thanking my mentor and adviser Dr. Alex 'Sandy' Pentland. He was kind enough to accept me to be part of his group and work under his supervision. In the past two years, he taught me many new things not in academia but also in life. He is also one of my biggest supporter and did not hesitate to write me recommendation and support letters. I would like to dedicate my first thank you to him. I know that it is not enough for all the help I have received for the past two years, and I look forward to continue working with him for my PhD.

Of course without the help of my amazing committee, this thesis would have not been possible. I would like to thank Dr. Burçin Bozkaya and Dr. Thomas Hardjono for their help and patience with me. Their feedback was always greatly appreciated and they were both generous with their time and advices.

The Human Dynamics group at the MIT Media Lab has been a family that supported me from day one. I would like to thank all my colleagues and collaborators many of whom are or were at the Human Dynamics group. I would like to specially thank Alejandro Noriega Campero, Abdullah Almaatouq, Dr. Peter Krafft, Dhaval Adjodah, Oren Lederman, Stephen Buckley, Julie Hall, Julia Schachnik and Theresa Hagen.

I would like to thank Dr. Erhardt Graeff for helping me find this amazing place and for introducing me to Ethan Zuckerman who is an amazing mentor and friend. Ethan welcomed me into his Center for Civic Media in 2015 and it was a great experience that has impacted my life in many positive ways. I would like to thank him for his time and also thank all my friends at the Center for Civic Media.

Additionally, I would like also to thank Joichi "Joi" Ito, Keira Horowitz and the entire Media Lab team for providing me with guidance and help for the last two years. I want to also thank the dean of the School of Architecture and Planning Dr. Hashim

Sarkis and his amazing assistant Mrs. Peggy Cain for their help and guidance.

In addition, I would like to thank Dr. Filip Cuckov who is my long life mentor, my undergraduate adviser and my friend.

I would like to extent my appreciation and gratitude to my sponsor Kuwait University, and I would like to specially thank Dr. Jehad Al Dallal for his tremendous support. I would like also to thank the hiring committee at the Information Science Department for putting their trust in me. In addition, I would like to extend my appreciation to Dr. Fawaz Al-Anzi, Dr. Mohammad Al-Failakawi . Dr. Abdulatif Alkhulaifi, Dr. Adel Alhusainan, Dr. Abdullah Almutairi and Samira Almansour.

Additionally, I would like to thank Kuwait Foundation for Advancement of Science (KFAS) and specially Dr. Adnan Shihab-Eldin, Dr. Lubna Okasha, Dr. Mohammad Salman and Mr. Faisal Almatrouk. I would like to thank the former director of National Bureau for Academic Accreditation and Education Quality Assurance (NBAQ) in Kuwait Dr. Nouria Alawadhi for supporting my case.

Furthermore, I would like to take this opportunity to thank Eng. Noura Alghurair for believing in me when I was at Kuwait Institute for Scientific Research (KISR). She was always challenging me to become a better researcher and without her support I would have not been able to push myself to become a better researcher and a better human being.

Of course, I can not forget the help and support that I have received from all my friends here at the lab and back home. I would like to thank Eng. Mohammad Almeer, Eng. Abdullah Alkhulaifi, Eng. Abdullah Alhusainan, Eng. Burhan Khaled and Eng. Abdulaziz Alsafar.

At the end, I could not finish this section with out thanking the most important person in my life my mother, Amal Alwesais. She is the only person in my life that I could not thank her enough; I will not be able to repay what she has done for me even if I spent every second of my life trying.

Finally, I would like also to thank my father, Saleh Alotaibi, for his advices and support at every stage in my life. Finally, I would like to thank my brother Abdulaziz and my two sisters Sarah and Nawal.

This Masters thesis has been examined by a Committee of Media Arts and Sciences program as follows:

Signature redacted

Dr. Burçin Bozkaya. . . . . .
Thesis Reader,
Professor of Business Analytics, Sabanci University

Signature redacted

Dr. Thomas Hardjono . . . . . .
Thesis Reader,
Technical Director at MIT Internet Trust Consortium, MIT

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Sharing data is one of the hardest problems specially these days. Many people raised many concerns about their privacy and their data that many companies collect. The most recent case is Facebook and Cambridge analytica before the USA presidential elections when they used people's private data to manipulate and influence them. My adviser Dr. Alex 'Sandy' Pentland back in 2007 predicted that such thing would be a reality in the near future [7]. Moreover, data and sensitive data in particular carry information that could be weaponized to target and manipulate people. In the recent years, data storage and differential privacy became essential fields of study. Wisdom of the machines (WOM) tries to address information sharing problem, and also tries to connect multiple different concepts together to arrive to a plausible solution.

The solution is heavily inspired by wisdom of the crowds (WOC) [12] and specially the learning that happens in a network setting. Because of my work in 'The Wisdom of the Network: How Adaptive Networks Promote Collective Intelligence' [20], I am going to borrow the same concepts and apply it to machine learning models thus the name wisdom of the machines.

Furthermore, this work extends on an ongoing research project at MIT Media Lab called Open Algorithms (OPAL). OPAL is a platform that manages sharing and accessing of data. This is currently achieved by breaking the problem into multiple different buckets, and has different user types with clearly defined roles. I will cover OPAL extensively in chapter 3.

Additionally, this work is also inspired by a team at Google. The team demonstrated a way to combine learning from multiple machine learning models[17]. They showed such collaborative learning could exist, and I will cover their approach in chapter 4.

At the end, I will build on top of the previous concepts to introduce WOM as a solution for information sharing using OPAL and extend OPAL to support machine learning model generation.

Chapter 2

Wisdom of the Crowd

In this chapter, I will cover the origins and the concept of wisdom of the crowds, and I will cover a famous paper that challenges the concept. Then, I will conclude by explaining my work and the experiments I did in this field.

Wisdom of the crowds (WOC) was introduced by Francis Galton a statistician who wanted to test the law of large numbers. He attended a competition where a group of farmers were estimating a weight of an ox. They submitted their answers using a paper, and the one closest to the correct weight would won a prize. After the competition, Galton collected eight hundreds independent answers. He then averaged all the answers to arrive to an answer that was closer to the correct weight of the ox than the winner's estimation. He published his findings in Nature back in 1907 Galton [12].

Later, James Surowiecki published a book titled 'The Wisdom of Crowds' in 2004 [21] where he covered Galton's paper and he tried to explain the concept. He cited Galton's work and gave a good explanation to the phenomena. His work inspired many researchers to study the phenomena closely and come up with alternative explanations, and one famous study that argued against WOC was by Jan Lorenzo et al. [16]. They argued that the phenomena could be explained easily by the herding effect. People submit answers close to each other because they do not want to be outliers. People usually want to be close to the truth, but also not far from others.

They used an experiment to show the herding effect in their study. They asked a

group of students six different questions about Switzerland and neighboring countries. The total number of participants was 144 students. All of them were students at Eidgenössische Technische Hochschule Zürich in Zurich, Switzerland. In the experiment, they had three different conditions or treatments 'no information', 'aggregate information' and 'full information'. In 'no information' treatment, subjects relied only on their knowledge and did not receive any information from their peers. This treatment was used as a control to the entire experiment. The second treatment group 'aggregate information', participants were asked a question and they would submit an initial answer. After that, they received an arithmetic mean of all 12 initial answers, and they were asked to submit an answer. They had the opportunity to submit the same answer and they repeated that five times for each question. The last group 'full information', subjects receive a plot with all the other participants answers. They were asked to submit five answers to each question, and they received an update from their peers after each round. The following two plots summerizes the results 2-1.

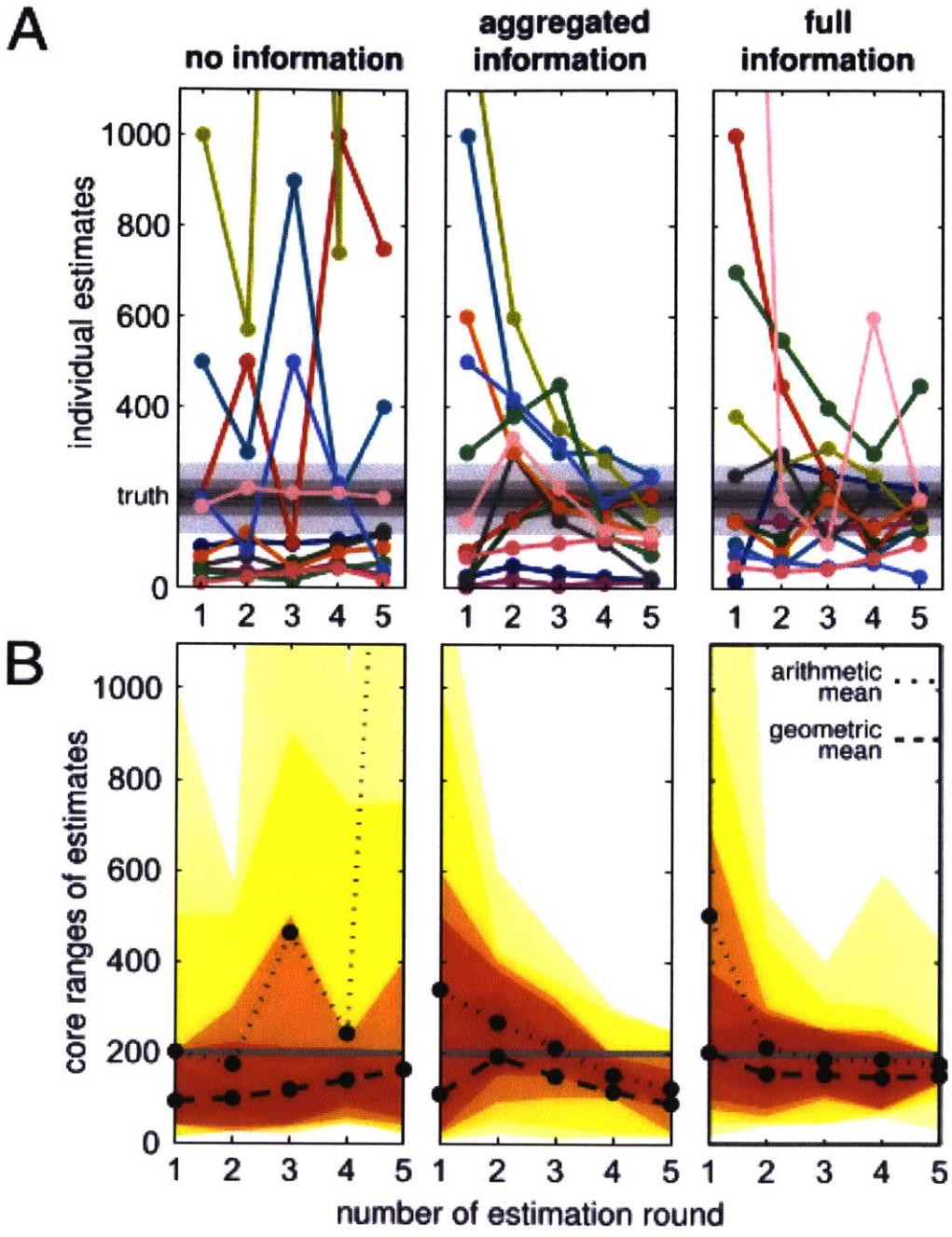

Figure 2-1: A and B show the individual and core ranges of estimates in the three different treatment groups to one of the questions.

Many studies came after Lorenz et al. [16] paper that argued against their findings,

and one study in particular found that all individuals who participated in the social part of the experiment 'aggregated information' and 'full information' made less errors than 'no information' [10]. Farrell [10] used the same data that Lorenz et al. [16] conducted, and he was able to produce the following figures 2-2. His work among others inspired me and others to work on alternative explanation to this phenomena and this is why we came up with a new alternative study to explain this phenomena from a network perspective [20].

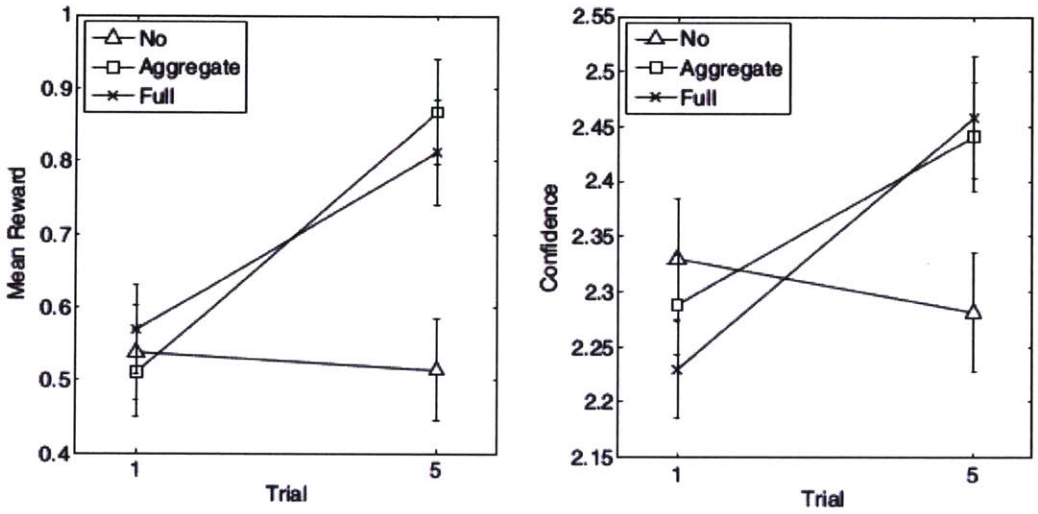

Figure 2-2: Both figures show that participants made the same errors at round 1, but only in the 'aggregated information' and 'full information' treatments were able to reduce the errors by round 5.

2.1 Wisdom of the dynamic network

In this paper Noriega-Campero et al. [20], we presented our hypothesis that WOC exists because of people's ability to aggregate information. We granted the existence of WOC phenomena to multiple skills that people have. We think that the most important skills are the aggregation ability and the ability to forget or block information from propagating. We simulated the blocking of information by allowing individuals to select who to follow and unfollow in an experiment. We conducted an online ex-

periment, and we had three different treatment conditions. Finally, we showed that the group who participated in last treatment made less errors on average than the other studied groups in the paper.

We had three different treatments or groups control group, static network group and dynamic network group. We ran the experiment on Amazon Mechanical Turk [22], and employed real human subjects to participate in this experiment. We had more than seven hundred unique participants, and we built a online platform that used Websockets [11] to allow participants to pass data in real-time between them. The experiment is modeled after guess the correlation game [3] where participants guessed the correlation of a scattered plot on a slider. All plots were positively correlated and they had three different levels of difficulty easy, medium and hard. All participants played twenty rounds, and we switched the difficulty of the plots on round ten to simulate a shock in the system. The participates were placed randomly in one of the treatment groups and they had different stages in each round depending on their treatment. Figure 2-3 shows the different stages for each treatment group and shows a sample of the different plots.

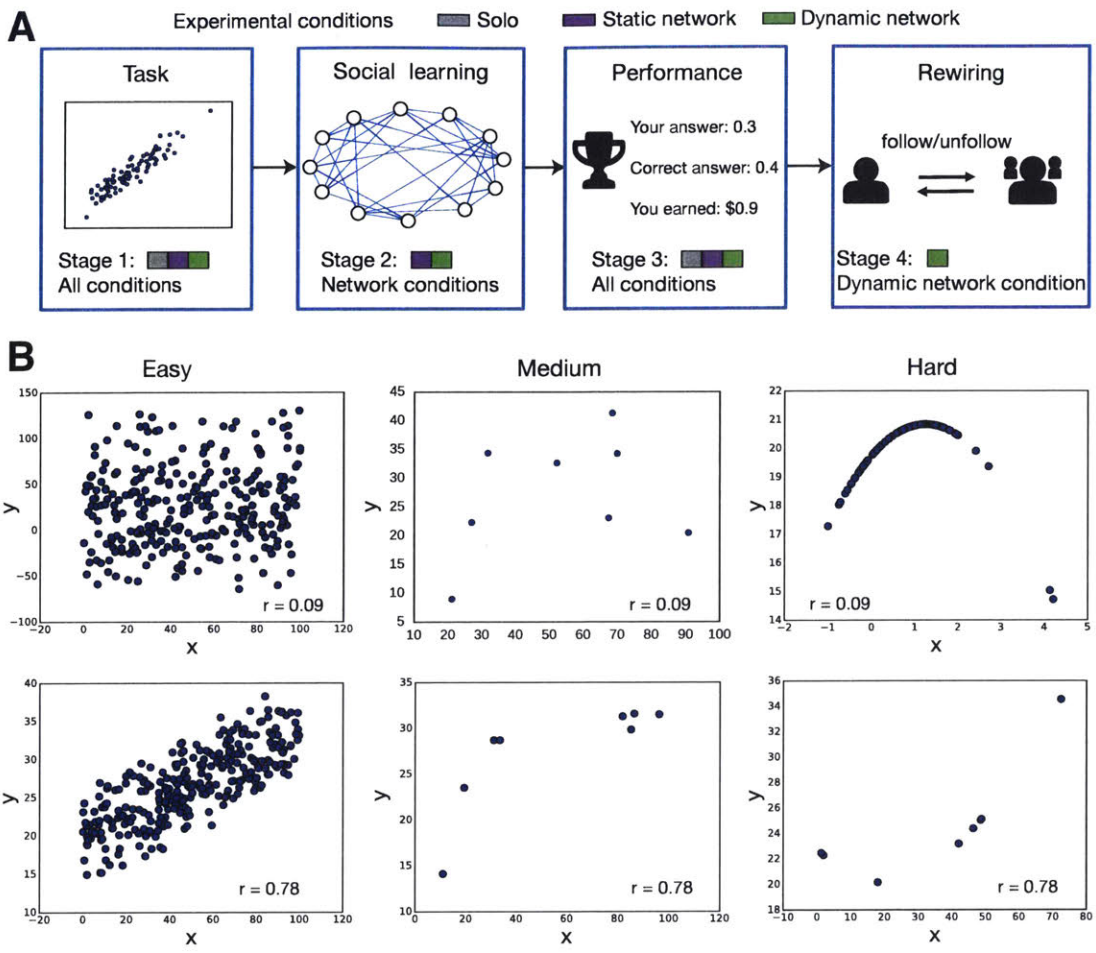

Figure 2-3: Row A shows the different stages of the game. Row B shows an example of the different plots and different difficulty levels.

First treatment was the control group or solo treatment, and it is a version of the game where participants would play the game without any social interactions. This version modeled the traditional WOC where participants submitted estimations independently and relied on their learning skills to preform better. Players in this treatment started by reading the instructions of the game. Then, we had a comprehension test to see if they actually understood the task. After passing the test, players saw a plot and a slider and were asked to submit their estimations. Then, the player continued to a page where we reveled the correct answer to them and told them how much they made in the previous round. After that, they proceeded to round two

which was similar to round one. They submitted a total of twenty different answers to different plots. At the end, we collected their feedback about the experiment.

Second treatment was the static group, and in this version of the game players started by going to the instructions pages similar to the first treatment. Then, they took a comprehension test about the task. After passing the test, players progressed to a waiting page. This game had a real-time interaction feature and we wanted all players to be in the same stage at all times. Each lobby had a capacity of twelve players, and once all players were at the waiting page, the game started by showing the first plot. The game placed each player at a random location in a random network with in-degree and out-degree of three. Each round had three different stages. In initial stage, players saw a plot and submitted independent guesses. In revised guess stage, players saw the same plot as in the initial stage and they were asked to submit another guess. We enabled them to "interact" with their peers in their social network. This means, players got notified in real-time when their peers changed their answers. After submitting for the second time, they went to the final stage where we revealed the correct answer and how much they made at the previous round. After finishing all twenty rounds, they went to a survey page to collect extra information and feedback from them. All questions were optional to answer as in the previous treatment.

Last treatment was the dynamic group, and in this version of the game players started by going through the instruction pages which were similar to the previous two treatments. Each round had three stages, and the stages are a copy of the static treatment except in the last stage. In the last stage, players saw how much each player made at the previous round and they had the chance to change their position in the network for the next round. The game ended with a survey to collect feedback.

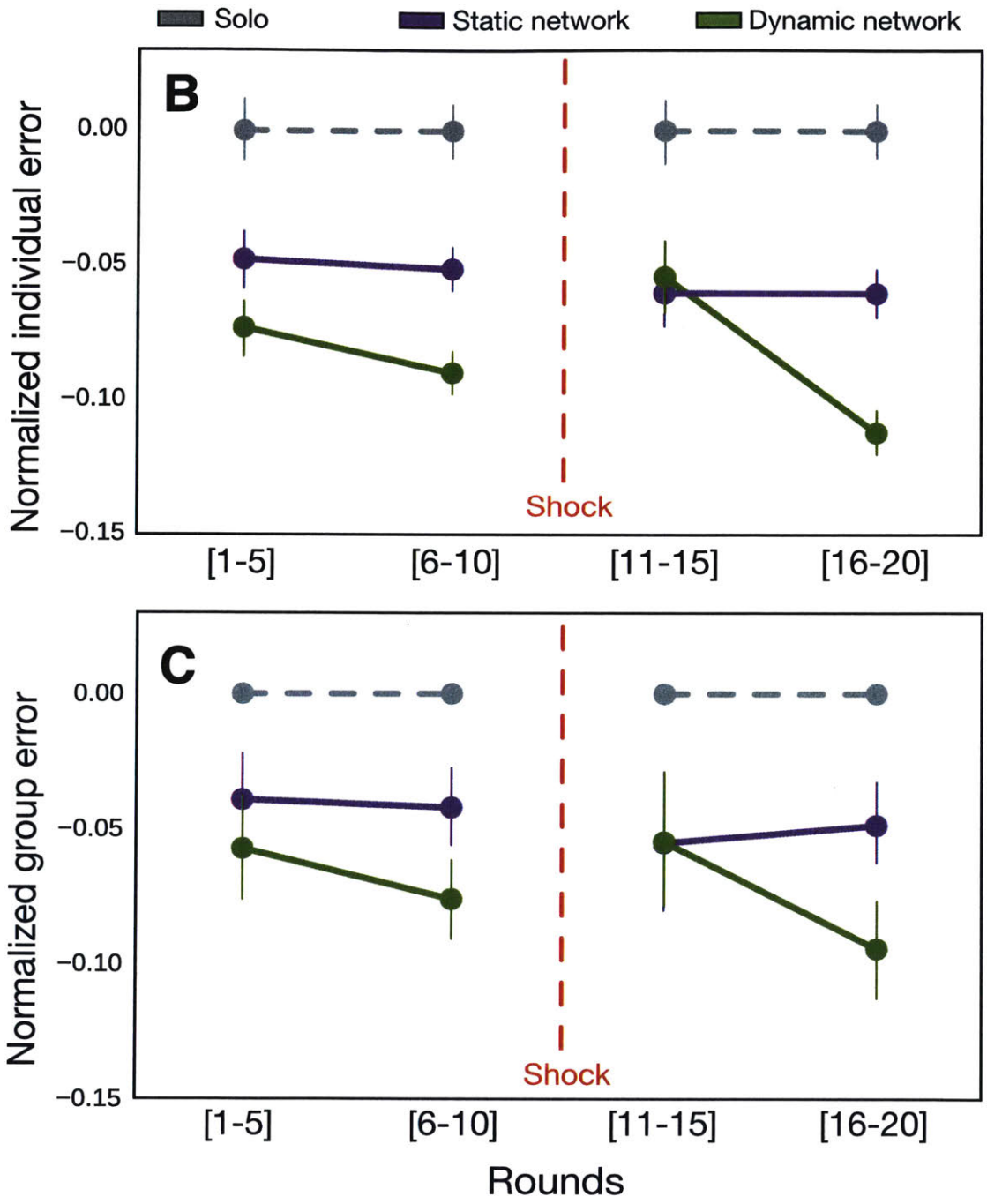

Figure 2-4: Plot B shows different individual errors for different rounds of the game, and plot C shows the group error. The dynamic group had the least aggregated error in both plots

After running this experiment, we ran multiple analysis to understand which treatment had the most effect. Our initial hypothesis was that the dynamic treatment group would preform better than the static treatment group, and the static treatment group would preform better than the control treatment group. We were correct in our assumption original assumption, and we were surprised to see that all individuals in the dynamic treatment group preformed better than the best individual in the static treatment. The same goes for the static treatment and the control group treatment. the dynamic treatment group had less errors and their standard deviation was more centralized.

The dynamic group was able to evolve their network to find a good network structure that minimized the average error. People in this group where not always following the best performers, but they were following the best social information aggregators. The following two figures show that not only the best performers where in the dynamic group but also they show that it impacted the whole group performance.

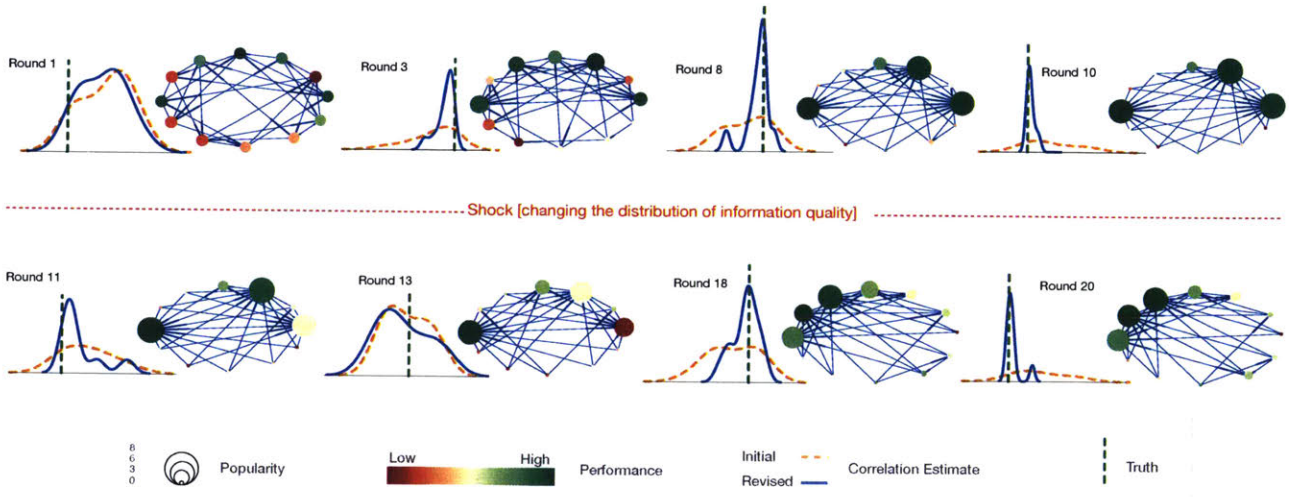

Figure 2-5: This figure shows multiple variables of a one run of the experiment. It shows the network configuration, it also shows the true answer, the initial guesses distribution and the revised guesses distribution. We can see that by round 10 and 20, almost every one was submitting the same answer and they all were centralized around the true answer.

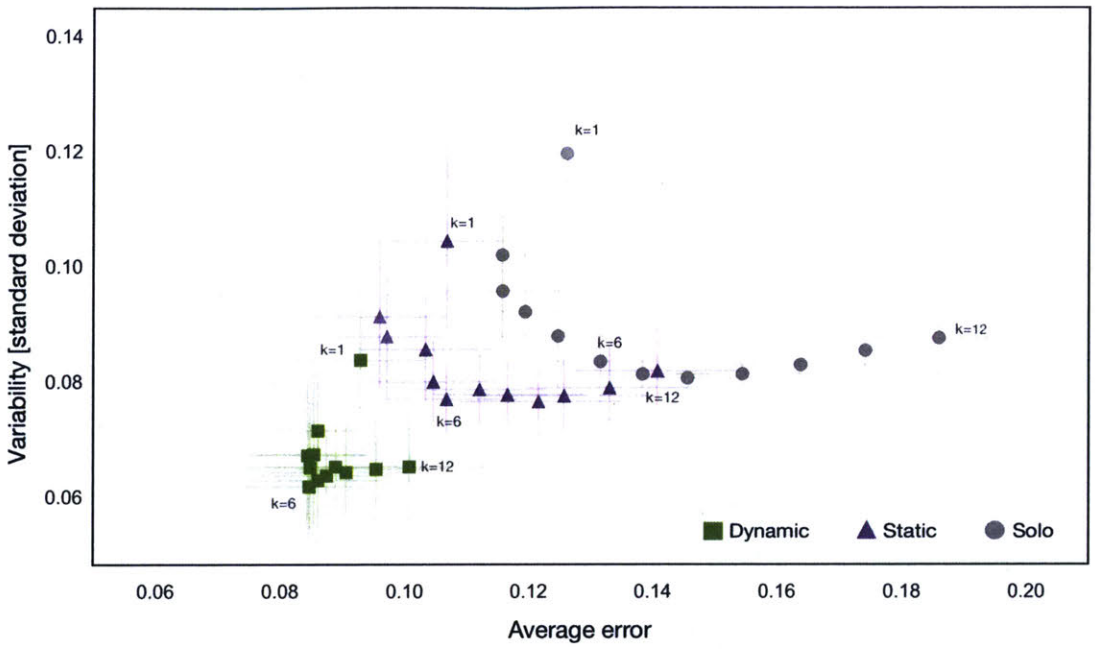

Figure 2-6: In this figure, we show the best twelve players in terms of low average error and low standard deviation. We can see that the best players are in the dynamic treatment group.

Chapter 3

Open algorithms (OPAL)

In this chapter, I will cover Open Algorithms project (OPAL). I start by sharing a brief history of the problem that OPAL is trying to solve. I then cover the current status and functions of OPAL. I conclude this chapter by demonstrating a use case of OPAL.

Sharing data securely is one of the hardest problems specially if the data contains sensitive information. There are many situations when two parties want to share information securely and privately. This is a classical problem in the field of computer science and cryptography. There are many environments where cryptographic encryption is required e.g. sharing data through the Internet or storing sensitive data on disk. The problem becomes complex when scientists or entities want to use the data without compromising its privacy. To illustrate the complexity of the problem, I will cover Netflix competition when they released a subset of their users' data to researchers. Netflix asked them to design and build a new movie recommender system. Netflix anonymized the released data and replaced users' names with unique identifiers. However, a group at University of Texas showed that using this anonymized data set they were able to find individuals' name. This was accomplished by combining Netflix's data with IMDB open data. They published their results in the following paper [19], and they were able to identify health information for a few clients.

Furthermore, homomorphic ecnryption and differential privacy try to solve this problem. Homomorphic encryption is a field that allows individuals to run useful

computations on encrypted data without the need to decrypt the data. This method enables using untrusted third party clouds and servers to run heavy computations without risking the security of the data. It also allows to get useful answers from multiple data providers by using multiparty computations. On the other hand, differential privacy is a way to preserve privacy by limiting the number and type of questions one's might ask on a particular data set. Apple announced recently that they started using differential privacy to protect the privacy of their clients. The method of differential privacy is essentially different from homomorphic encryption where the first one introduces noise and other statistical tricks to provide the queirer with useful answers while protecting the raw data.

Additionally, de Montjoye et al. [8] marked the birth of OPAL as a plausible solution to the data sharing problem. The following 3-1 shows all different components of OPAL. OPAL tries to shift the data sharing paradigm by differentiating between users and their roles. Each user or role type has a specific job, and OPAL manages these jobs to provide safe and correct answers. From the figure below, we can see Data provider, Vetted Algorithm, Blockchain and querier. I will cover each in the following sections.

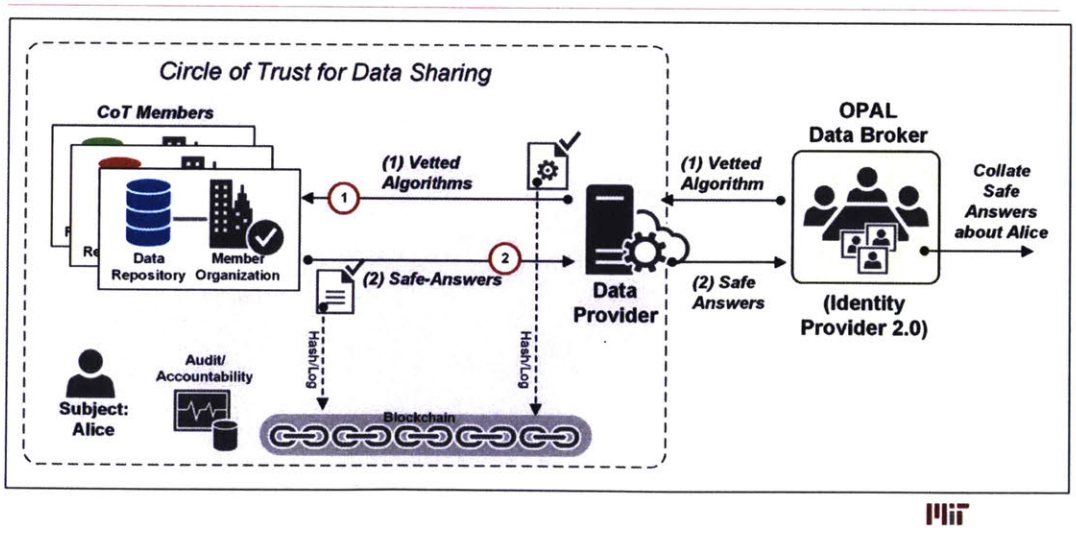

Figure 3-1: This figure shows all OPAL's components.

3.1 Data providers

Data providers are the most critical part of the platform. They provide the platform with data sets and configure the meta data for each data set. Each data they list on OPAL has to be assigned to a schema, and each schema has a schema provider. OPAL uses this design to know how to retrieve data correctly.

Data providers could share their data with everyone on OPAL or limit the access to the data to certain individuals. They also could share data drivers and keep their data in their databases. OPAL then uses the drivers to retrieve answers without downloading the data. Figure 3-2 shows a list of data sets on OPAL, and figure3-3 shows how to add new data set to OPAL.

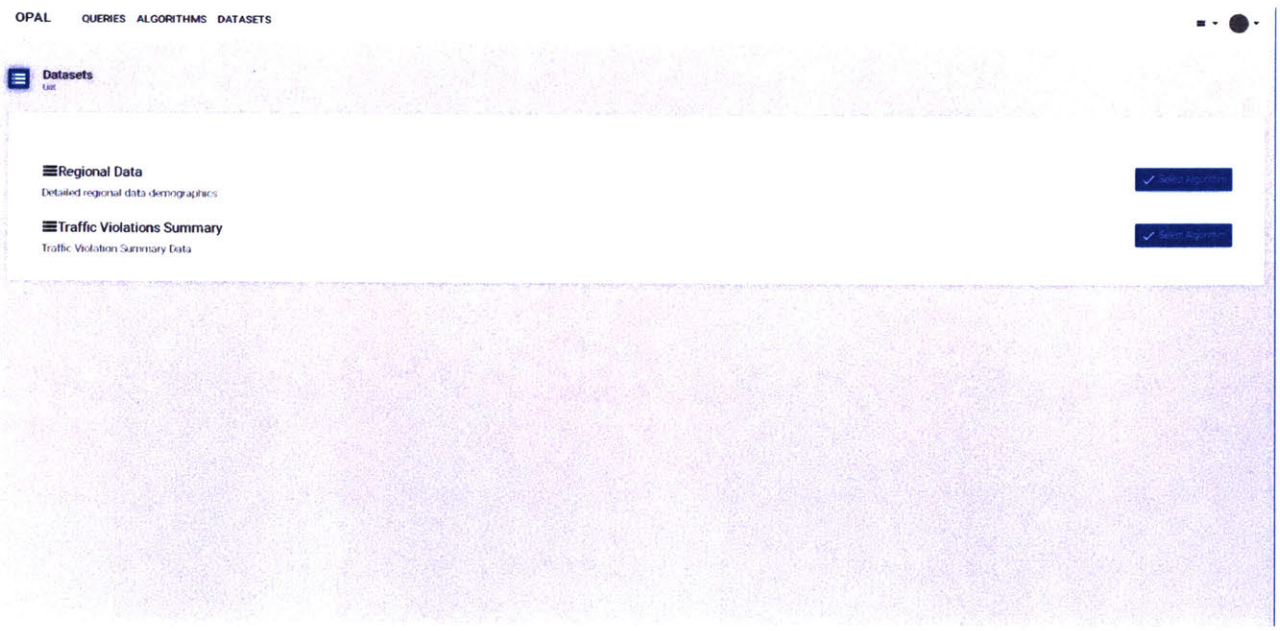

Figure 3-2: This is an example of data sets [6]. Here, there are two data sets that can be used, and on the top right corner we can connect a data set with an algorithm.

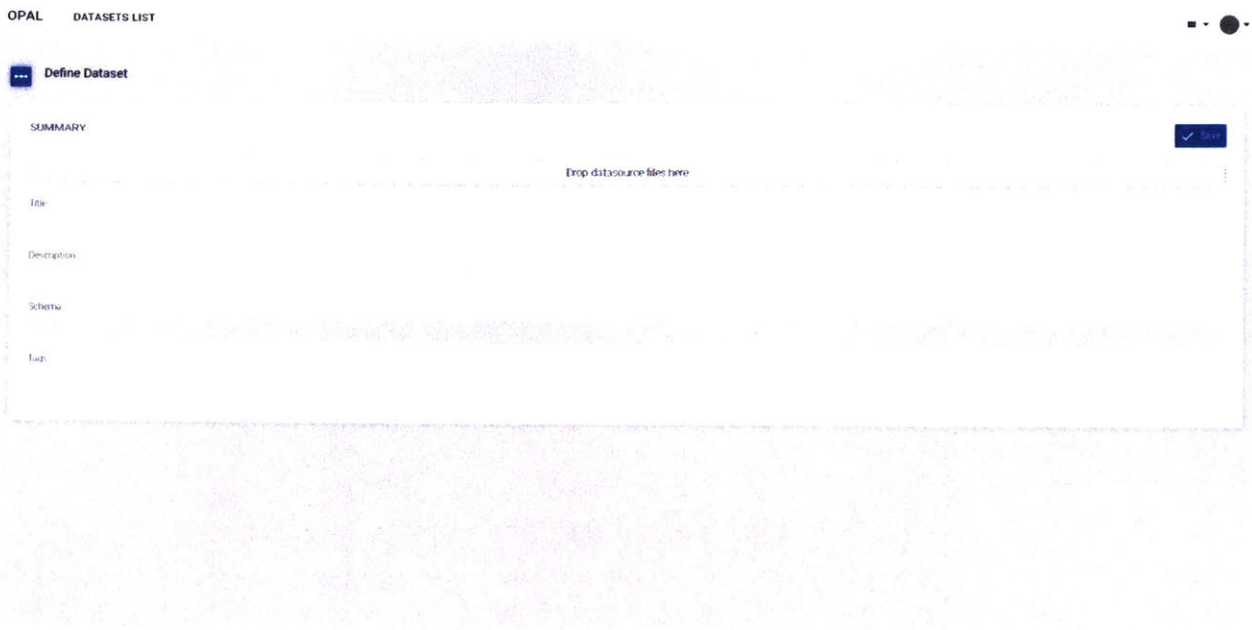

Figure 3-3: This page shows the required fields to add a new data set to OPAL. In this example, the data provider needs to provide a title, description, schema and tags.

3.1.1 Schema providers

The schema providers are related to two different core components of OPAL data sets and algorithms. This is how OPAL can filter out algorithms that can not operate on certain data sets. When data providers list new data sets, they need to link them to schemas. Similarly, when algorithm providers list new algorithms, they need to connect them to schemas. OPAL does a simple check before the executing each query, and only execute the query if both schemas match. This approach allows clients to have multiple data sets with the same data schema, and thus OPAL could run the same query on all data that have the same schema. Figure 3-4 shows a list of schemas.

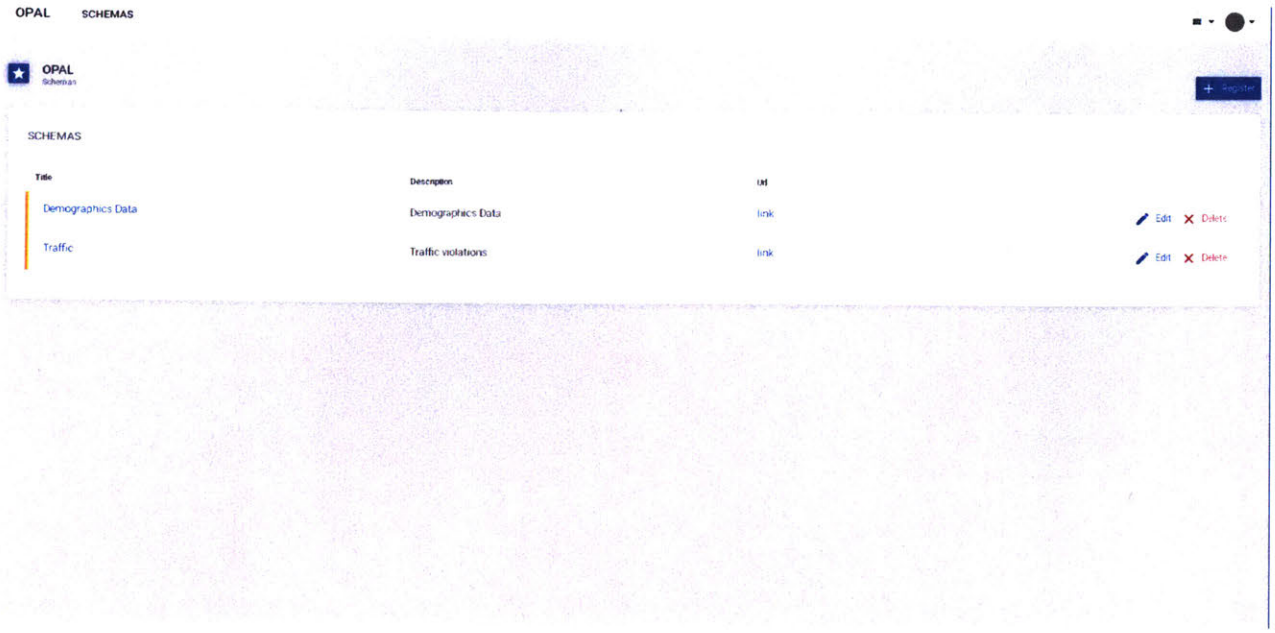

Figure 3-4: This is an example of of schemas on OPAL website [6].

3.2 Algorithm providers

This is the second most important user type on OPAL. Algorithm providers list their algorithms on OPAL. Providers can add their algorithms and link them to a specific schema. The algorithms usually operate on a particular data schema. OPAL uses the data driver to send the algorithm back to the data and structures the data to fit the schema. OPAL also makes sure that both the algorithm schema and the data schema match.

Algorithm providers are usually scientists or companies who build mathematical models. These models predict certain behavior and use the data as a prior. After the algorithm provider submits a new algorithm, it goes into a vetting process. This is a manual process to make sure that the algorithm is fair and does not leak any information about the data. Figure 3-5, figure 3-6 and figure3-7 show the steps required to list and add a new algorithm.

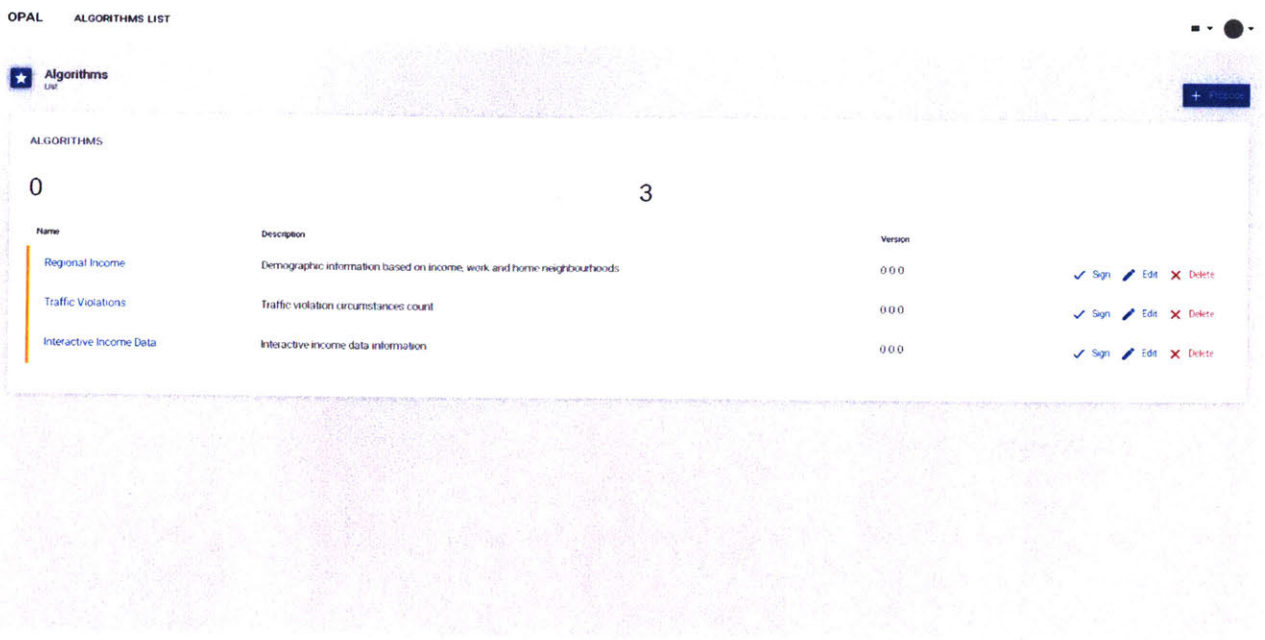

Figure 3-5: In this page, there are a list of algorithms. There are descriptions for each algorithm and version number. In addition, the algorithm provider can submit an algorithm to be listed.

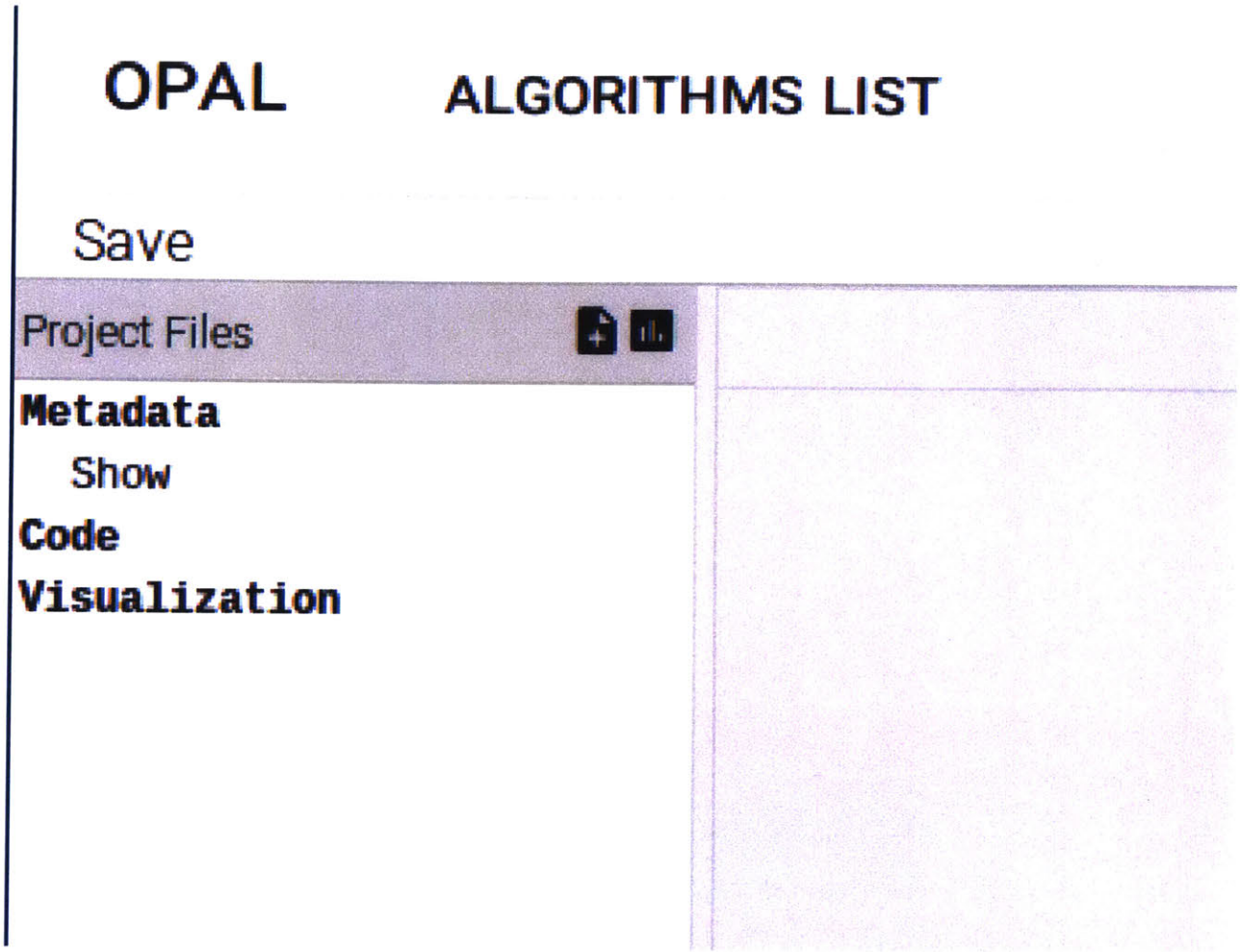

Figure 3-6: This page is where the client can propose another algorithm to be added to OPAL. On the side, there is a list of tabs for meta data of the algorithm, code and visualization. The algorithm provider must fill out the meta data and adds the code to this tab. The visualization is an extra feature that makes it easy to visualize the analysis.

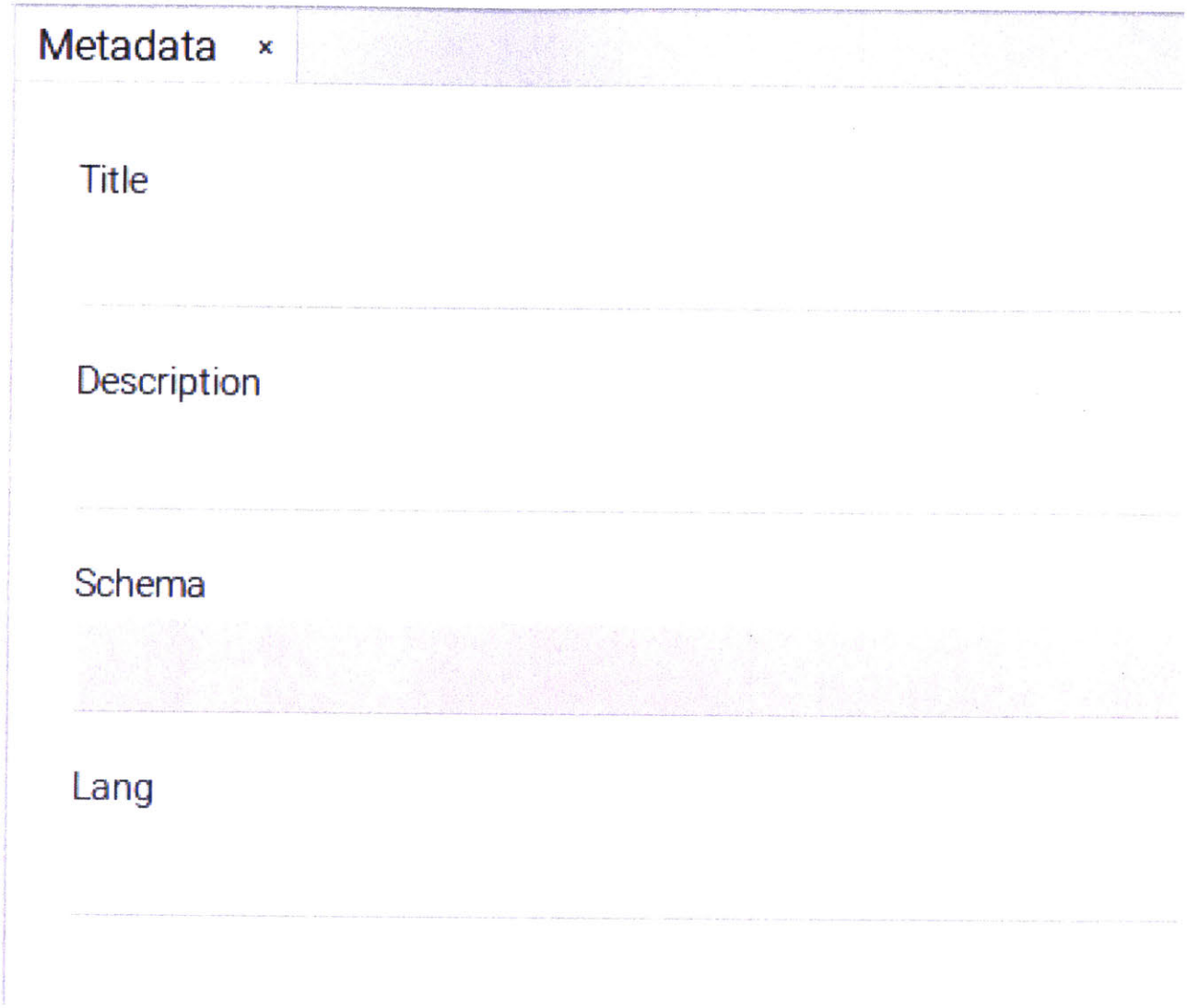

Figure 3-7: This figure shows the required information for each algorithm to be filled. Here is also where algorithm providers connect the algorithm to a schema.

3.3 Queriers

This is the last user type on OPAL. Queriers are the commissioners of queries and they connect a data set or data sets with an algorithm. They use OPAL ecosystem and select an algorithm and one or multiple data sets to be executed on. OPAL checks if they have the right permissions on the algorithm and the data sets as well. OPAL

then runs the algorithm on each data set using its driver. Later, OPAL aggregates the answers and generates a response file. This file then gets shared with the querier.

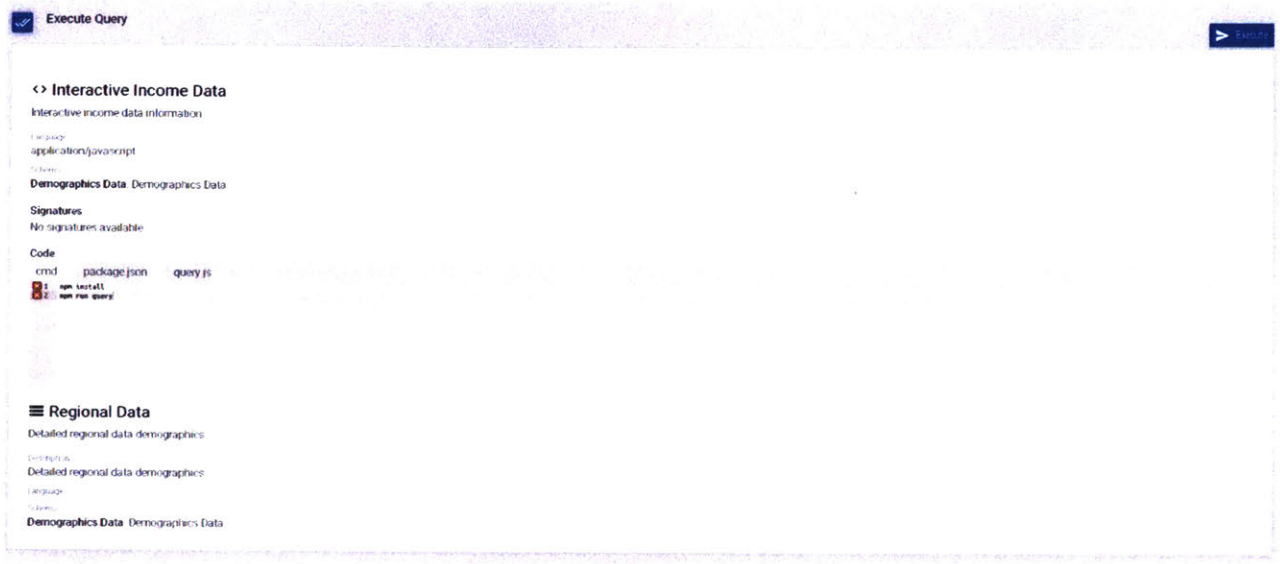

Figure 3-8: This figure shows the related information to execute a query. The query consists of a vetted algorithm and a data set. The code section has all the information related to the algorithm. Some algorithms require certain packages to be installed before they can run. The tab 'cmd' has all the commands that this algorithm requires before running on the data. In this case, the algorithm would execute on a data titled "Regional Data".

Figure 3-9: This is a list of all packages that this algorithm requires to be installed before running.

Code

cmd | package.json | query.js

```
1  const fs = require('fs')
2  const csv = require('fast-csv')
3
4  const ageMap = ['10-15', '15-20', '20-25', '25-30', '30-35', '35-40', '40-45', '45-50', '50-55', '55-60', '60-65', '65-70', '70+'];
5
6  function readCSVFile(filename) {
7      var stream = fs.createReadStream(filename);
8      return new Promise((resolve, reject) => {
9          let result = [];
10         return csv
11             .fromStream(stream, {ignoreEmpty: true})
```

Figure 3-10: This figure shows a snippet of the code. The code has been vetted and digitally signed by a verifying entity which reviewed the code and made sure that it does not leak nor discriminate against any point in the data.

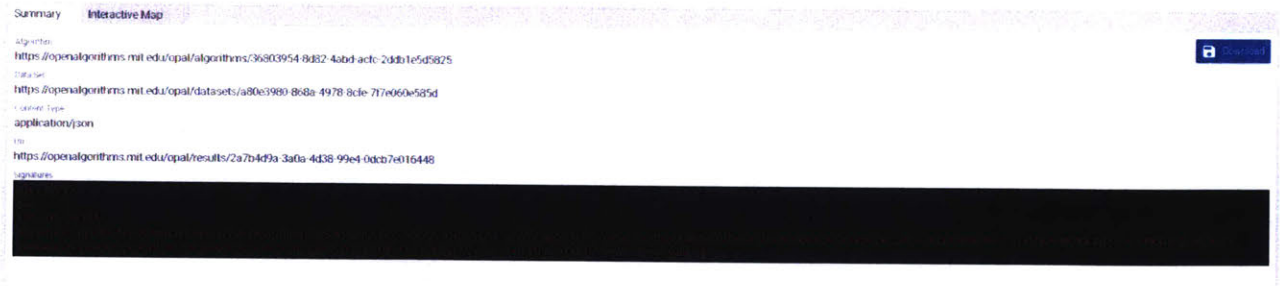

Figure 3-11: This figure shows the output after executing the algorithm on the data. We can see that this run has a digital signature and this signature is going to be used to log this particular transaction. At the top right corner, we can see a download button, and this button would download the generated data. The data generated could be aggregated from multiple sources.

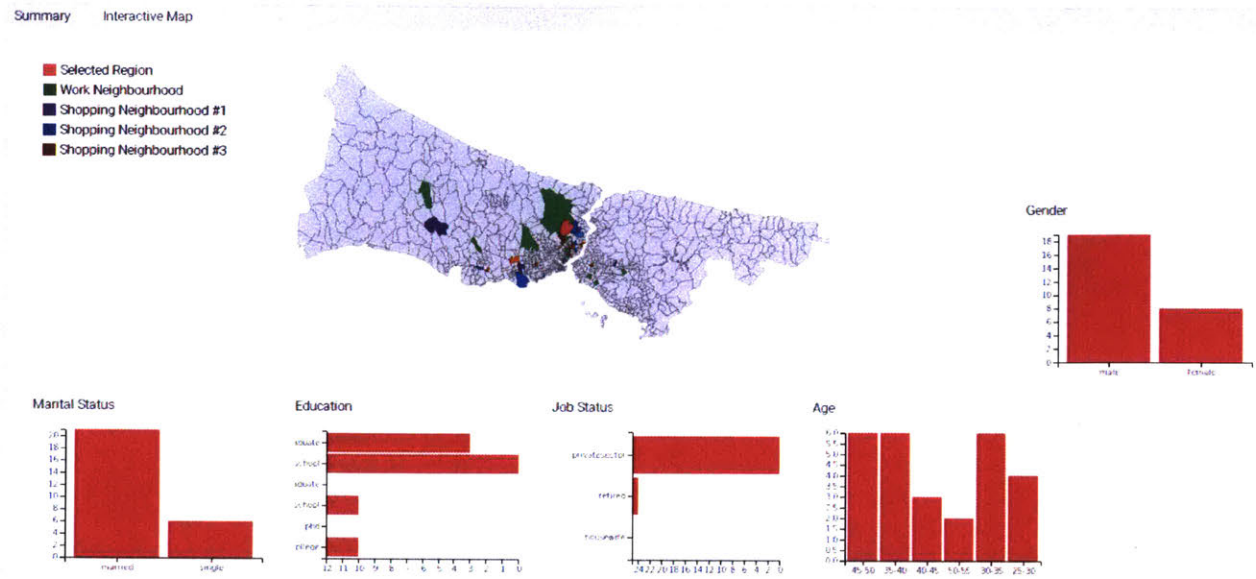

Figure 3-12: This figure shows a visualizations of the analysis which was provided by the algorithm provider when they submitted their algorithm to OPAL.

3.4 Blockchain

OPAL like any other solution is not prone to bugs and issues. This is why OPAL comes with a blockchain integration component. The blockchain is used to log all the transactions on OPAL platform. This adds another level of security to OPAL because unlike other systems where they keep the code and logs on the same server. OPAL shares all its logs on a public blockchain, but it also could integrate easily with a private blockchain. Currently, we are working on a version to integrate OPAL with Ethereum. This makes OPAL useful specially for governments when they want to share data with their citizens but also maintain a logs.

3.5 Current OPAL functions

Currently, OPAL provides answers after they have been aggregated by the executor i.e. data provider. These answers could be an average of certain feature or summation

over an entire feature, inter alia. There are many limitations with the current OPAL version and this these focuses on adding machine learning to the next version of OPAL. I will cover a plausible solution that fits with the current implementation of OPAL in chapter 5.

Chapter 4

Federated Learning

This chapter will discuss a paper that a team at Google publish recently. I will start by talking briefly about the problem and summarize their findings.

A team at Google wanted to improve Android users' experience. They designed a new keyboard application for the new version of Android, and this keyboard has a words prediction feature. There are many keyboards with the same exact feature. However, they changed the approach of how this keyboard stores and shares users' data. The other keyboards relied on a centralized server to provide predictions for each word. This is an issue, because any input from the user needs to be logged and shared back with a server. These inputs could be passwords or other sensitive information. The team came up with a new way to share information across users and allowed for a better user experience.

The solution was to have two different machine learning training processes one on phones and another on a server. The first machine learning model which is stored locally on each Android device trained on the users' input. This model would learn words that users used, and stored a local copy of the words on each device. This local copy never got shared with the outside world. However, hyperparameters or the weights of the local model got shared back with the server. The server aggregates all hyperparameters from all devices to build universal prediction engines one for each language that Android keyboard supported. This method generated a model that had all the information from all small models. The server then sent the generated

model back to the Android devices to update their local machine learning models. This method was introduced by McMahan et al. [17].

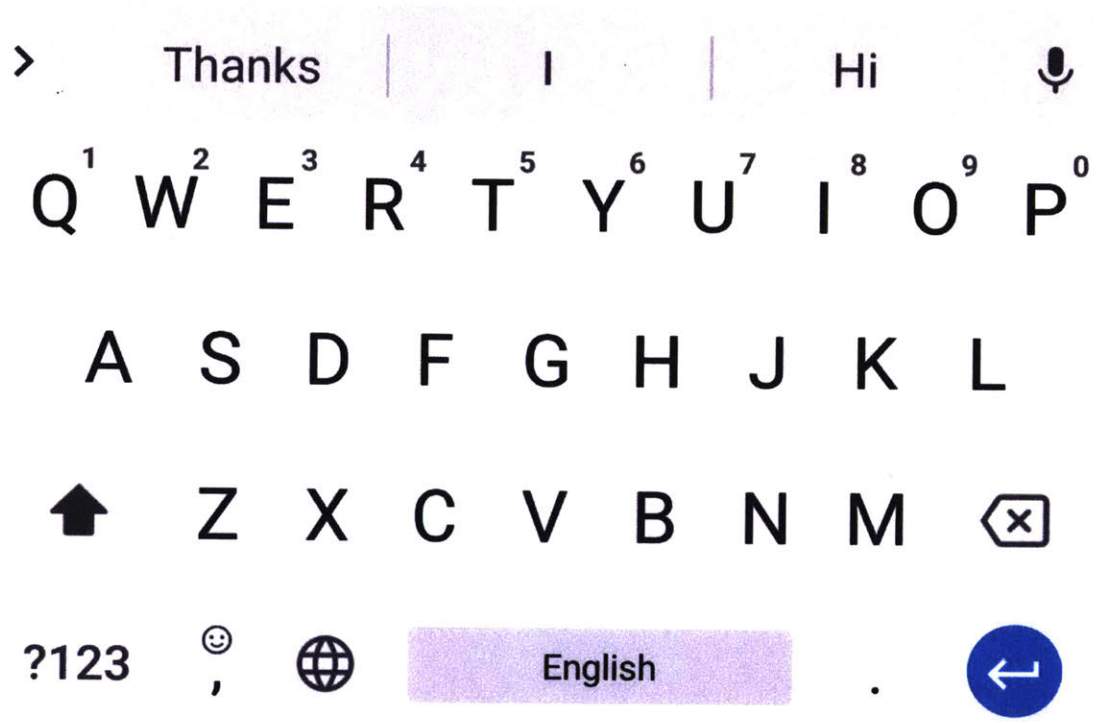

Figure 4-1: Android keyboard with word suggestion feature. In this example, it shows "Thanks", "I" and "Hi" as a possible word choices.

The team wanted this solution to have the least amount of communication with the server. This approach is meant to be used on mobile devices where wireless network is not available all the time, and also expensive compared with regular internet access. One clear assumption as well was the feature set has to be the same across different models. Figure 4-2 illustrates how federated learning works.

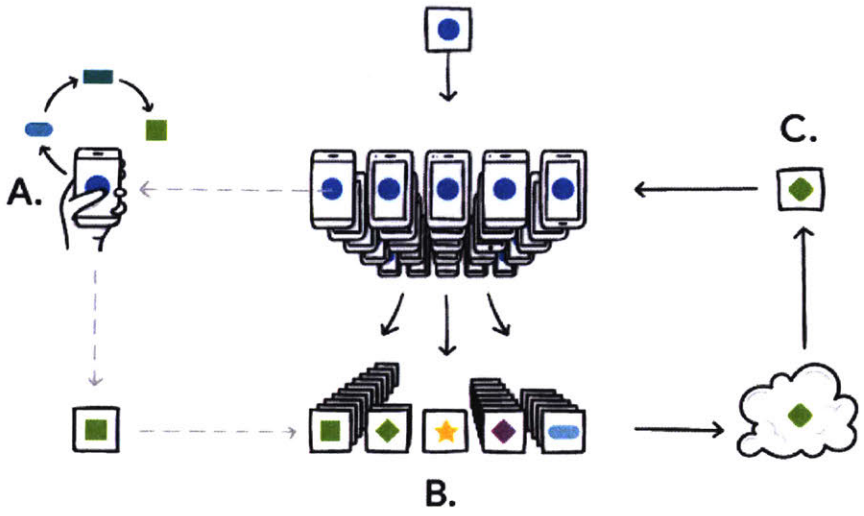

Figure 4-2: Federated learning illustration [2]. 'A' is where the model learns from the user. 'B' is where other users share their hyperparameters and 'C' is where the cloud computes the new model.

McMahan et al. [17] showed that it is possible to combine different models under the stated assumption where the models were trained on the same features. In this paper, they used MNIST data [15], and they tried two different ways to split the data across clients. The first method, they shuffled the entire data and partitioned the data after. The second method, they sorted the data based on the label and then they partitioned the data. They also used two different machine learning models and at the end of the paper they demonstrated that this method also worked using another data set.

The method relied on knowing the number of clients and the number of data points each client had on their local storage. Later, they asked each client to train on their local data set and to reach a certain accuracy. Then, they aggregated the hyperparameters to normalize them by the number of total clients. They summed all normalized hyperparameters into one machine learning model. The final model had all the information from all small models.

Additionally, they tested their system with different number of clients or agents. They used the following numbers of agents 1, 10, 20, 50 and 100. They recorded how each group preformed. The first case is just the normal approach of training an agent where, and this agent generates the generalized model by training on all data. The other four cases is where they tested their algorithm by training the agents on a subset of the data once by sorting on the digit labels and another by shuffling the entire data set before splitting them equally. The table 4.1 summarizes the results of the paper.

	Shuffled data		Sorted data	
Number of agents	$B = \infty$	$B = 10$	$B = \infty$	$B = 10$
1	387	50	1181	956
10	339	18	1100	206
20	337	18	978	200
50	164	18	1067	261
100	246	16	NA	97

Table 4.1: B is the size of the batch at each iterations and $B = \infty$ means all the local data set. All agents were trained on a CNN [14] with two 5x5 convolution layers. All agents where trained on epoch equals to five and trained until the final model reached an accuracy of 99%. The columns show number of rounds required to reach that level.

Chapter 5

Wisdom of the Machines

This chapter builds on the concepts that I talked about in the last three chapters. The first chapter I talked briefly about wisdom of crowds and how a group of people making a decision might reach a better and more accurate decision than independent individuals. The second chapter covered OPAL and how this platform helps and manages sharing data between different parties. In the previous chapter, I covered a new way to train machine learning models using a concept called federated learning. In this chapter, I will introduce a new concept called the wisdom of the machines (WOM) which tries to combine these concepts together. This new method will make OPAL supports and generates machine learning models as safe answers instead of aggregated answers.

In WOC, we saw that a group of individuals can make independent estimates based on their private information. These estimates then get shared with the rest of the group or crowd, and then the group would aggregate all the estimates to arrive to a better answer. This is the core feature that I am going to borrow from this concept to add it to WOM which is the ability to aggregate independent estimates to generate a new and better estimate.

Similarly, federated learning uses different devices or users that make independent actions. They then train and modify their own machine learning models. These models get shipped back to a central server to train a new and better model. This model incorporate all the learning that each small model had.

OPAL splits the data sharing problem into multiple different users data providers, algorithm providers and queriers. I have already described what each user type does and I would like to introduce a new user type called model providers. Model providers propose models to be implemented on the OPAL. These models could be shared across different data sets and they could be connected to a data set using schema provider. I used the schema provider as a checking method because this is the method OPAL uses to make sure that algorithms and data are compatible. Next, I will talk about a prototype that I have built to demonstrate these concepts.

At the beginning, I used MNIST data set, and I implemented a network with N-number of agents. All agents are connected to one centralized node or a star shape network. Each node had a local data set to train on and the central node aggregated the hyperparameters from all the nodes in the network.

The models were implemented on OPAL, and all agents used OPAL's data provider layer to access their local data sets. OPAL made sure that each agent had the right level of permissions to access the data, and it used a specific data driver to load the data into the agents' memory. Next, the agents used their data to train their local machine learning model. The local machine learning model had a two layer CNN similar to the Keras MNIST tutorial model [5]. The central node had the same model as well. I chose the epoch time to be five and trained the models with the entire local data they had. After each training period, the agent would evaluate the accuracy of the produced model. If the model reached 99% of accuracy, the agent would stop the training and sent the hyperparameters back to a central node. The central node normalized the hyperparameters before adding them together. Next, it used the new aggregated hyperparameters as weights for its layers, and ran the evaluation function to evaluate the accuracy of the final model. Next, I generated a file containing the hyperparameters to be used as the answer for the query. The user then used the hyperparameters to generate the final machine learning model.

To demonstrate the previous method, I generated 6 different networks. Each network had different number of agents and I wanted to measure the accuracy of the final model and the number of training steps required to reach 99% in the training

period. I chose a network with 1 agent as the baseline for the following experiments. I had 5 other different configurations 2, 4, 6, 8, and 10 agent networks. Each experiment used MNIST data set and the data was shuffled and splitted equally between agents.

The following two figures 5-1 and 5-2 show the results from the experiments. Figure 5-1 shows that the final model preformed better as I increased the number of agents. I used the first model with 1 agent as a baseline, and there is a slight dip in accuracy after aggregating from 6 and 8 different agents. Figure 5-2 shows the average number of training required for each agent to reach a training accuracy of 99%. This figure shows that as I increased the number of agents, the more training steps were required. This makes sense because I was of the way I was partitioning the data. It is true that first model required less training operations, but it took more time to reach the desired accuracy.

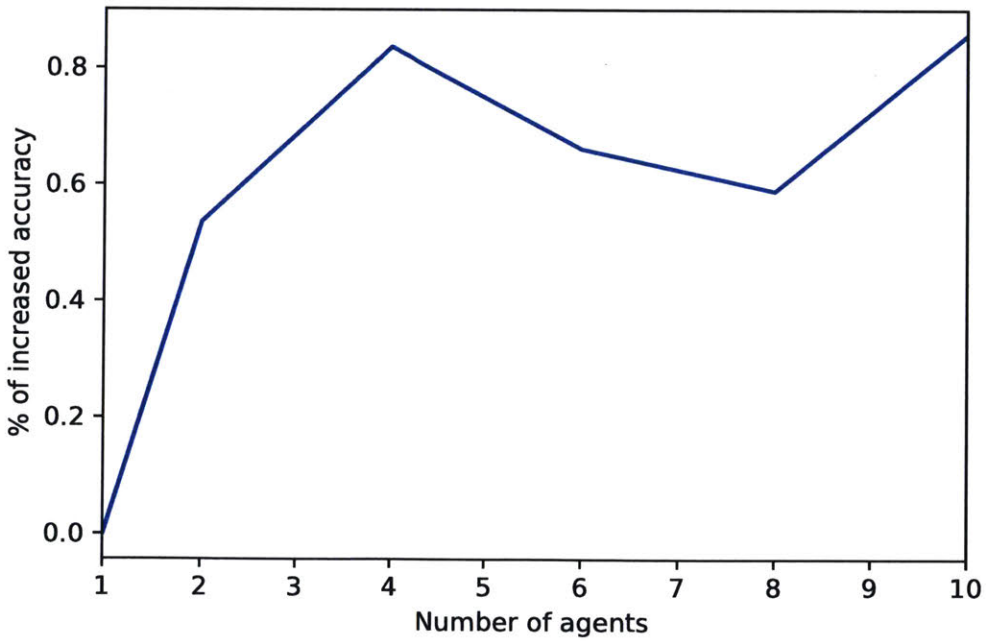

Figure 5-1: This is show the increase of accuracy of the final model. The base line is a model with one agent.

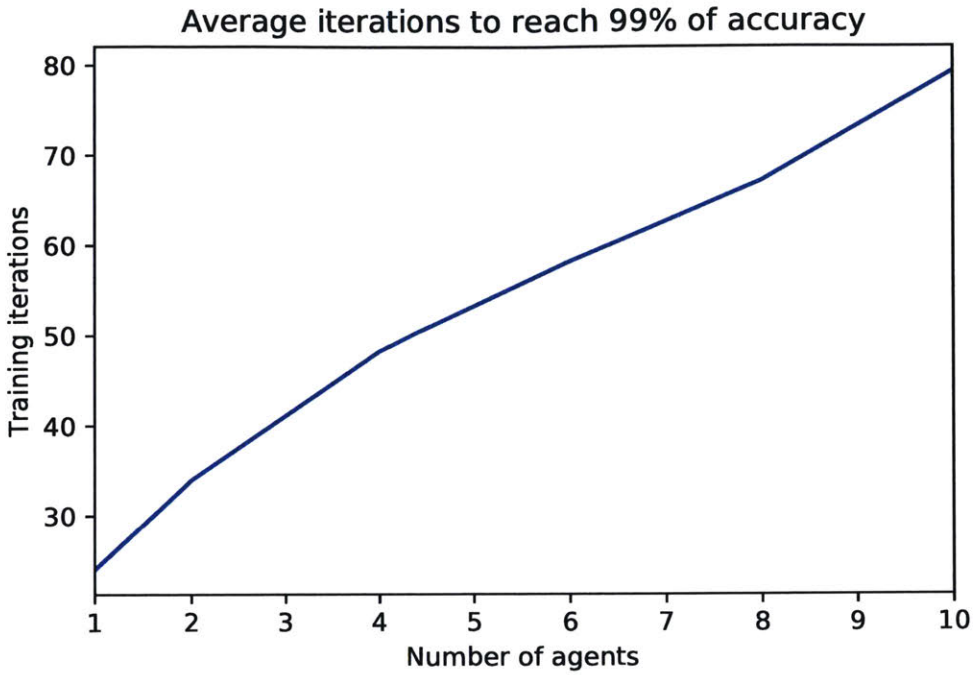

Figure 5-2: The average number of training required for each agent to reach an accuracy of 99%. It is linear relationship because the more agents in each configuration, the less the data they had.

5.1 Case study

There are many examples where this feature would be useful. The clear example is when two hospitals wanted to predict mortality rates but non of them wants to release their patients' data. This is a classic problem that is common in the medical field, and no one has a clear answer to it.

This is a great example to demonstrate how WOM would solve. The first step is for both hospitals to become data providers on OPAL. Then, they would list their data on OPAL by providing a data driver. Both hospitals need to agree on a set of common features. A model provider would propose a machine learning model for this problem. Then, experts from both hospitals would approve and vet this model. After they both signed the model, the model could be used on the data. The querier and in

this case could be the county where these two hospitals are could execute a query to know the mortality rates in that region. OPAL would train a machine learning model on each independent data set and aggregate the hyperparameters to the querier. The querier could download a machine learning model that could predict the mortality rates.

This method could be extended to allow for many use cases as well. It will open up many possibilities and boundaries as well. This method could allow industries where they do not normally share data to share information freely.

Chapter 6

Conclusion

In this thesis, I started by describing the concept of WOC and where it came from. Then, I gave an example of how it is being used today. I then talked about the importance of aggregators that allow for such phenomena to exit.

Additionally, I talked about the importance of OPAL and how difficult the problem it is trying to solve. I described each individual component of the system and demonstrated how it worked using real world example. I concluded by covering the current limitation of OPAL.

Furthermore, I covered a recent publication from a team at Google. The team demonstrated a way to build machine learning models. They wanted independent agents to share these models and not run into a risk of data privacy. They used a new method to improve a new version of Android keyboard application.

At the end, I combined these concepts to cover a new feature of OPAL. This feature used all the concepts to extend OPAL and added machine learning capability to OPAL. This is a great feature and will open up many possibilities for OPAL.

Appendix A

OPAL demo code

A.1 Cmd

```
npm install
npm run query
```

A.2 Packages

```
{
  "name": "query",
  "version": "1.0.0",
  "description": "",
  "scripts": {
    "query": "node query.js"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "fast-csv": "^2.4.0"
  }
}
```

A.3 Query

```
const fs = require('fs')
const csv = require('fast-csv')

const ageMap = ['10-15', '15-20', '20-25', '25-30', '30-35', '35-40', '
    40-45', '45-50', '50-55', '55-60', '60-65', '65-70', '70+'];

function readCSVFile(filename) {
    var stream = fs.createReadStream(filename);
    return new Promise((resolve, reject) => {
        let result = [];
        return csv
            .fromStream(stream, {ignoreEmpty: true})
            .on("data", function (data) {
                result.push(data);
            })
            .on("end", () => {
                resolve(result)
            })
    })
}

function increment(host, key, value) {
    if (!host[key][value]) {
        host[key][value] = 0;
    }

    host[key][value]++;
}


function calcSpend(host, key, trans, value) {

    let transVal = parseInt(trans),
        val = parseInt(value);

    if (transVal && val && (transVal > 0 || val > 0)) {
```

```
36          let ref = host[key];
37          if (!ref) {
38              host[key] = ref = {trans: 0, spend: 0};
39          }
40          ref.trans += transVal;
41          ref.spend = ref.spend > 0 ? (ref.spend + val) / 2 : val;
42      }
43  }
44
45
46  let areaData = {};
47  Promise
48      .all([
49          readCSVFile("./data/userInfo_10K.csv"),
50          readCSVFile("./data/userPerformance_10K.csv"),
51          readCSVFile("./data/userTransInfo_10K.csv")
52      ])
53      .then(data => {
54          let [userInfo, performance, transInfo] = data;
55
56
57          let shopData = transInfo.reduce((memo, item)=> {
58              memo[item[0]] = item;
59              return memo
60          }, {});
61
62
63          let userRisk = performance.reduce((memo, item) => {
64              memo[item[0]] = item;
65              return memo;
66          }, {});
67
68          //Area specific histograms
69          userInfo.forEach(item => {
70              let [userId, gender, marital, edu, job, income, age, homeNei
    , workNei] = item,
```

```
71
72              area = areaData[homeNei],
73              ageKey
74
75          //Build histograms
76          if (!area) {
77              areaData[homeNei] = area = {
78                  gender: {},
79                  marital: {},
80                  edu: {},
81                  job: {},
82                  age: {},
83                  churn1: 0,
84                  churn2: 0,
85                  risk: 0,
86                  workNei: {},
87                  homeSpend: {trans: 0, spend: 0},
88                  workSpend: {trans: 0, spend: 0},
89                  shopSpend1: {},
90                  shopSpend2: {},
91                  shopSpend3: {},
92              }
93          }
94
95          let risk = userRisk[userId];
96
97          if (risk && !area.workNei[workNei]) {
98              area.workNei[workNei] = 1;
99          } else {
100             area.workNei[workNei]++;
101         }
102
103         let shop = shopData[userId];
104
105         if (shop) {
106             let [
```

```
107                     duserID,
108                     numTransHomeNei,
109                     medSpendHomeNei,
110                     numTransWorkNei,
111                     medSpendWorkNei,
112                     numTrans5411,
113                     medSpend5411,
114                     numTrans5691,
115                     medSpend5691,
116                     numTrans5541,
117                     medSpend5541,
118                     numTrans5812,
119                     medSpend5812,
120                     nei1,
121                     nei2,
122                     nei3] = shop;
123
124
125             calcSpend(area, 'homeSpend', numTransHomeNei,
    medSpendHomeNei);
126             calcSpend(area, 'workSpend', numTransWorkNei,
    medSpendWorkNei);
127
128
129             calcSpend(area.shopSpend1, nei1, numTrans5411,
    medSpend5411)
130
131
132             calcSpend(area.shopSpend2, nei2, numTrans5691,
    medSpend5691)
133
134
135             calcSpend(area.shopSpend3, nei3, numTrans5541,
    medSpend5541)
136
137         }
```

```
138
139             //Calculate area churn/risk
140             if (risk) {
141                 area.churn1 += parseInt(risk[1])
142                 area.churn2 += parseInt(risk[2])
143                 area.risk += parseInt(risk[3])
144             }
145
146
147             let ageIndex = Math.round(age / 5) - 2;
148             if (ageIndex > ageMap.length - 1) {
149                 ageIndex = ageMap.length - 1;
150             }
151
152             ageKey = ageMap[ageIndex];
153
154             if (ageKey) {
155                 increment(area, 'gender', gender)
156                 increment(area, 'marital', marital)
157                 increment(area, 'edu', edu)
158                 increment(area, 'job', job)
159                 increment(area, 'age', ageKey)
160             }
161
162         });
163
164         fs.writeFile('result.json', JSON.stringify(areaData), () => {
165             console.log("Done")
166         });
167
168     });
```

Listing A.1: Javascript code that analyses the data.

A.4 Client class

```
1 #!/usr/bin/env python3
```

```
2 import pickle
3
4
5 class Client:
6     num = 0
7
8     def __init__(self, model, features=None, labels=None):
9
10         self.model = model
11         self.score = None
12
13         self.id = Client.num
14         Client.num += 1
15
16         if features is not None and len(features) and len(labels):
17             assert len(features) == len(labels), 'Features and labels
    sizes does not match'
18             self.features = features
19             self.labels = labels
20
21     @property
22     def weights(self):
23         return self.model.get_weights()
24
25     @weights.setter
26     def weights(self, values):
27         self.model.set_weights(values)
28
29     def fit(self, epochs, batch_size):
30         if len(self.features) and len(self.labels):
31             assert len(self.features) == len(self.labels), 'Features and
    labels sizes doesn\'t match'
32             return self.model.fit(self.features, self.labels, batch_size
    =batch_size, epochs=epochs)
33         else:
34             raise Exception('The client does not have data')
```

```
35
36      def evaluate(self, x, y):
37          self.score = self.model.evaluate(x, y)
38          return self.score
39
40      def dump_model(self):
41          with open('{}.obj'.format(self.id), 'wb') as f:
42              pickle.dump(self.weights, f)
43
44      def load_model(self, file):
45          if self.model:
46              with open(file, 'rb') as f:
47                  self.weights = pickle.load(f)
48          else:
49              raise Exception('The client does not have a model')
```

A.5 Wisdom of the machines code

```
1  #!/usr/bin/env python3
2  from random import shuffle, sample
3  import os
4  import re
5  import argparse
6
7  import keras
8  from keras.datasets import mnist
9  from keras.models import Sequential
10 from keras.layers import Dense, Dropout, Flatten
11 from keras.layers import Conv2D, MaxPooling2D
12 from keras import backend as K
13
14 import numpy as np
15 from client import Client
16 from network_generators import generate_erdos_renyi_connected,
       generate_barabasi_albert_graph, generate_save_small_world
17
```

```
18
19 def data_partitioning(partitions, training_data, labels):
20     all_data = list(zip(training_data, labels))
21     size = int(len(all_data)/partitions)
22     shuffle(all_data)
23     return [all_data[(i*size):(size*(i+1))] for i in range(partitions)]
24
25
26 def create_cnn(input_shape, num_classes):
27     model = Sequential()
28     model.add(Conv2D(32, kernel_size=(3, 3),
29                      activation='relu',
30                      input_shape=input_shape))
31     model.add(Conv2D(64, (3, 3), activation='relu'))
32     model.add(MaxPooling2D(pool_size=(2, 2)))
33     model.add(Dropout(0.25))
34     model.add(Flatten())
35     model.add(Dense(128, activation='relu'))
36     model.add(Dropout(0.5))
37     model.add(Dense(num_classes, activation='softmax'))
38     model.compile(loss=keras.losses.categorical_crossentropy,
39                   optimizer=keras.optimizers.sgd(),
40                   metrics=['accuracy'])
41     return model
42
43
44 def reshape_test(test, rows, cols):
45     if K.image_data_format() == 'channels_first':
46         x_test = test.reshape(test.shape[0], 1, rows, cols)
47         input_shape = (1, rows, cols)
48     else:
49         x_test = test.reshape(test.shape[0], rows, cols, 1)
50         input_shape = (rows, cols, 1)
51
52     x_test = x_test.astype('float32')
53     x_test /= 255
```

```
54
55      return x_test, input_shape
56
57
58  def reshape_train(train, rows, cols):
59      if K.image_data_format() == 'channels_first':
60          x_train = train.reshape(train.shape[0], 1, rows, cols)
61          input_shape = (1, rows, cols)
62      else:
63          x_train = train.reshape(train.shape[0], rows, cols, 1)
64          input_shape = (rows, cols, 1)
65
66      x_train = x_train.astype('float32')
67      x_train /= 255
68
69      return x_train, input_shape
70
71
72  def modify_two_weights(func, w1, w2):
73      w = []
74      assert len(w1) == len(w2)
75      for i, arr in enumerate(w1):
76          w.append(func(arr, w2[i]))
77      assert len(w) == len(w1)
78      return w
79
80
81  def add_two_weights(w1, w2):
82      return modify_two_weights(lambda x, y: x+y, w1, w2)
83
84
85
86  def aggregate_federated(clients: [Client], master_node: Client):
87      weights = [c.weights for c in clients]
88      for w in weights:
89          n = [i / len(clients) for i in w]
```

```
90          add_two_weights(master_node.weights, n)
91
92
93  def main():
94      parser = argparse.ArgumentParser()
95      parser.add_argument('--models', type=str, default=None)
96      parser.add_argument('--number_of_clients', type=int, default=10)
97      parser.add_argument('--epoch', type=int, default=5)
98      parser.add_argument('--edge_prob', type=float, default=0.2)
99      parser.add_argument('--fraction_of_clients', type=float, default
        =1.0)
100     parser.add_argument('--hubs', type=int, default=3)
101     parser.add_argument('--k', type=int, default=3)
102     args = parser.parse_args()
103
104     batch_size = 128
105     num_classes = 10
106
107     # This is E in the paper
108     epochs = args.epoch  # 5 or 20
109     if args.models:
110         number_of_clients = len(os.listdir(args.models))
111     else:
112         number_of_clients = args.number_of_clients  # C
113     fraction_of_clients = args.fraction_of_clients
114     # local_batch_size = 100  # B
115
116     # input image dimensions
117     img_rows, img_cols = 28, 28
118
119     # the data, split between train and test sets
120     (x_train, y_train), (x_test, y_test) = mnist.load_data()
121     clients_data = data_partitioning(int(number_of_clients *
        fraction_of_clients), x_train, y_train)
122
123     x_test, input_shape = reshape_test(x_test, img_rows, img_cols)
```

```
124     y_test = keras.utils.to_categorical(y_test, num_classes)
125
126     clients = []
127
128     for d in clients_data:
129         x_data = np.array([x for x, _ in d])
130         y_data = np.array([y for _, y in d])
131
132         y = keras.utils.to_categorical(y_data, num_classes)
133         x, _ = reshape_train(x_data, img_rows, img_cols)
134
135         clients.append(Client(create_cnn(input_shape, num_classes), x, y
    ))
136     print('Total number of clients is: {}'.format(len(clients)))
137     if not args.models:
138         for c in clients:
139             for i in range(200000):
140                 score = c.fit(epochs, batch_size)
141                 if score.history['acc'][0] >= 0.99:
142                     print('Model {} reached 99% of accuracy after {}
    iterations'.format(c.id, i))
143                     c.dump_model()
144                     break
145             else:
146                 print('Model {} could Not reach 99% after 20000
    iterations'.format(i))
147                 break
148
149     else:
150         files = os.listdir(args.models)
151         hubs = sample(files, args.hubs)
152         master = sample(hubs, 1)
153         hubs.remove(master[0])
154
155         for i in range(len(files)):
156             clients[i].load_model('{}/{}'.format(args.models, '{}.obj'.
```

```
        format(i)))
157
158         print('loaded all hyperparameters')
159
160     master_node = Client(create_cnn(input_shape, num_classes))
161     aggregate_federated(clients, master_node)
162     master_node.evaluate(x_test, y_test)
163     print('Federated Score : {}'.format(master_node.score))
164
165
166
167 if __name__ == '__main__':
168     main()
```

Bibliography

[1] Apple differential privacy, 2017. https://machinelearning.apple.com/2017/12/06/learning-with-privacy-at-scale.html [Accessed: 7-10-2018].

[2] Federated learning: Collaborative machine learning without centralized training data, 2017. https://ai.googleblog.com/2017/04/federated-learning-collaborative.html [Accessed: 7-10-2018].

[3] Guess the correlation, 2017. http://guessthecorrelation.com/ [Accessed: 7-10-2018].

[4] Facebook and cambridge analytica: What you need to know as fallout widens, 2018. https://www.nytimes.com/2018/03/19/technology/facebook-cambridge-analytica-explained.html [Accessed: 7-10-2018].

[5] Cnn for MNIST using keras, 2018. https://github.com/keras-team/keras/blob/master/examples/ [Accessed: 7-10-2018].

[6] Opal website, 2018. https://openalgorithms.mit.edu/ [Accessed: 7-10-2018].

[7] The cambridge analytica data apocalypse was predicted in 2007, 2018. https://www.wired.com/story/the-cambridge-analytica-data-apocalypse-was-predicted-in-2007/ [Accessed: 7-10-2018].

[8] Y.-A. de Montjoye, E. Shmueli, S. S. Wang, and A. S. Pentland. openpds: Protecting the privacy of metadata through safeanswers. *PloS one*, 9(7):e98790, 2014.

[9] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer, 2006.

[10] S. Farrell. Social influence benefits the wisdom of individuals in the crowd. *Proceedings of the National Academy of Sciences*, page 201109947, 2011.

[11] I. Fette and A. Melnikov. Rfc 6455-the websocket protocol.(2011). *URL https://tools.ietf.org/html/rfc6455*, 2016.

[12] F. Galton. Vox populi (the wisdom of crowds). *Nature*, 75(7):450–451, 1907.

[13] C. Gentry and D. Boneh. *A fully homomorphic encryption scheme*, volume 20. Stanford University Stanford, 2009.

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[15] Y. LeCun, C. Cortes, and C. Burges. Mnist handwritten digit database. *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, 2, 2010.

[16] J. Lorenz, H. Rauhut, F. Schweitzer, and D. Helbing. How social influence can undermine the wisdom of crowd effect. *Proceedings of the National Academy of Sciences*, 108(22):9020–9025, 2011.

[17] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, et al. Communication-efficient learning of deep networks from decentralized data. *arXiv preprint arXiv:1602.05629*, 2016.

[18] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system,âĂİ http://bitcoin.org/bitcoin.pdf, 2009.

[19] A. Narayanan and V. Shmatikov. Robust de-anonymization of large sparse datasets. In *Security and Privacy, 2008. SP 2008. IEEE Symposium on*, pages 111–125. IEEE, 2008.

[20] A. Noriega-Campero, A. Almaatouq, P. Krafft, A. Alotaibi, M. Moussaid, and A. Pentland. The wisdom of the network: How adaptive networks promote collective intelligence. *arXiv preprint arXiv:1805.04766*, 2018.

[21] J. Surowiecki, M. P. Silverman, et al. The wisdom of crowds. *American Journal of Physics*, 75(2):190–192, 2007.

[22] A. M. Turk. Amazon mechanical turk. *Retrieved August*, 17:2012, 2012.